Supervised Metric Learning to Rank for Retrieval via Contextual Similarity Optimization

Christopher Liao (Presenter), Theodoros Tsiligkaridis, Brian Kulis

BOSTON UNIVERSITY

MIT LINCOLN LABORATORY

Metric Learning Overview

- Learn an embedding space where similar samples are close together and dissimilar samples are far apart
- Samples with the same label should be closer than samples with different labels
- Applications: product retrieval, person re-identification, vehicle re-identification, search by image
- Existing methods include contrastive, triplet, ranking, and classification losses

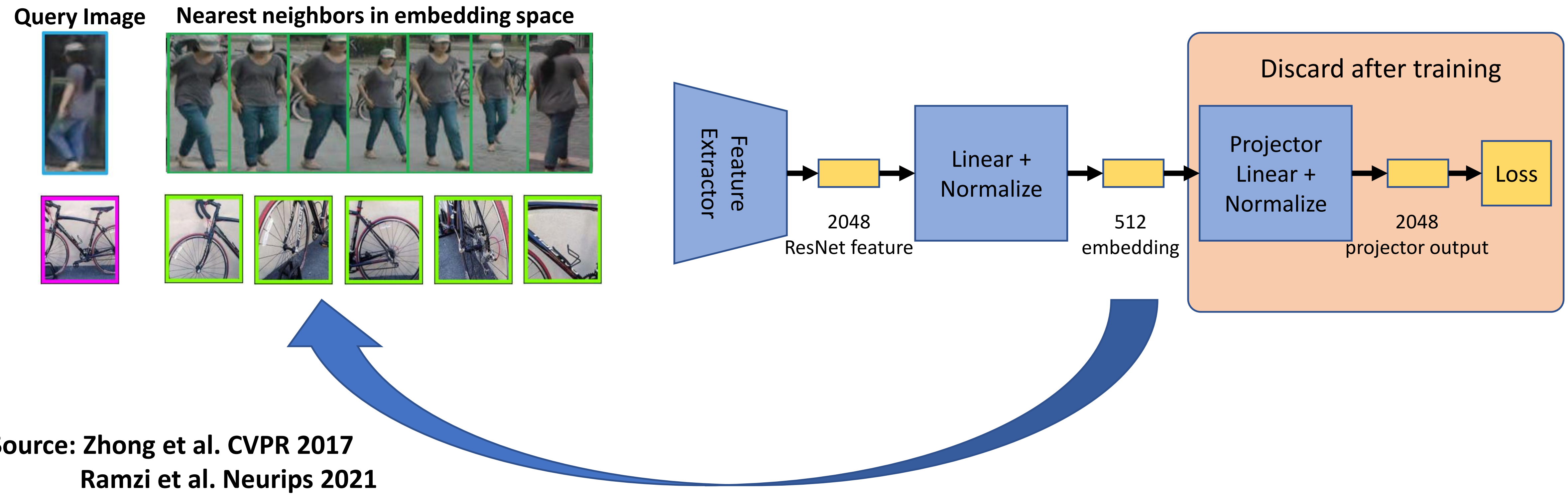
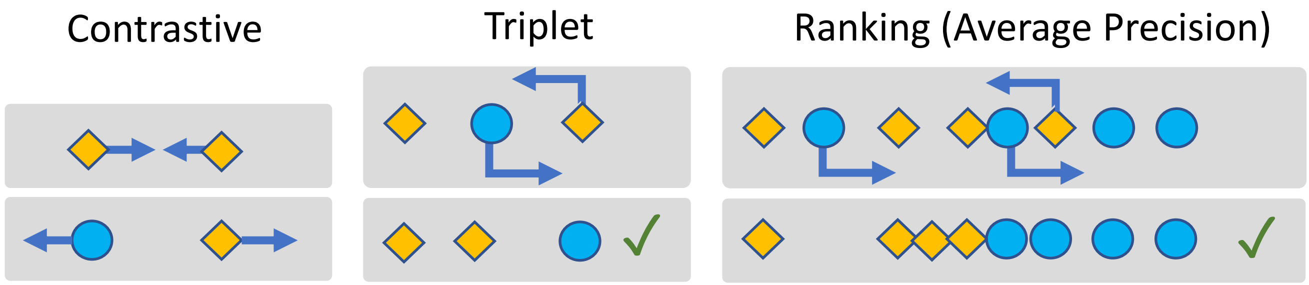


Image Source: Zhong et al. CVPR 2017
Ramzi et al. Neurips 2021

Metric Learning State-of-the-Art

- ***Contrastive***: Pull together similar pairs and push apart dissimilar pairs
  - *Disadvantage*: fixed margin values
- ***Triplet***: Make positive samples closer than negative samples
  - *Disadvantage*: triplet sampling is hard
- ***Ranking***: Rank positive samples closer than negative samples by maximizing average precision
- ***Classification***: Train a classifier then throw away the last linear layer
  - *Disadvantage*: Needs to be finely tuned. Not good on tasks with many labels



Problem with Metric Learning Datasets

Figure 1. Examples of metric learning labels which are inconsistent with semantic information from two standard benchmarks: CUB (top) and SOP (bottom). These labels are caused by a visual feature which is not present or barely visible.

Our Method Overview

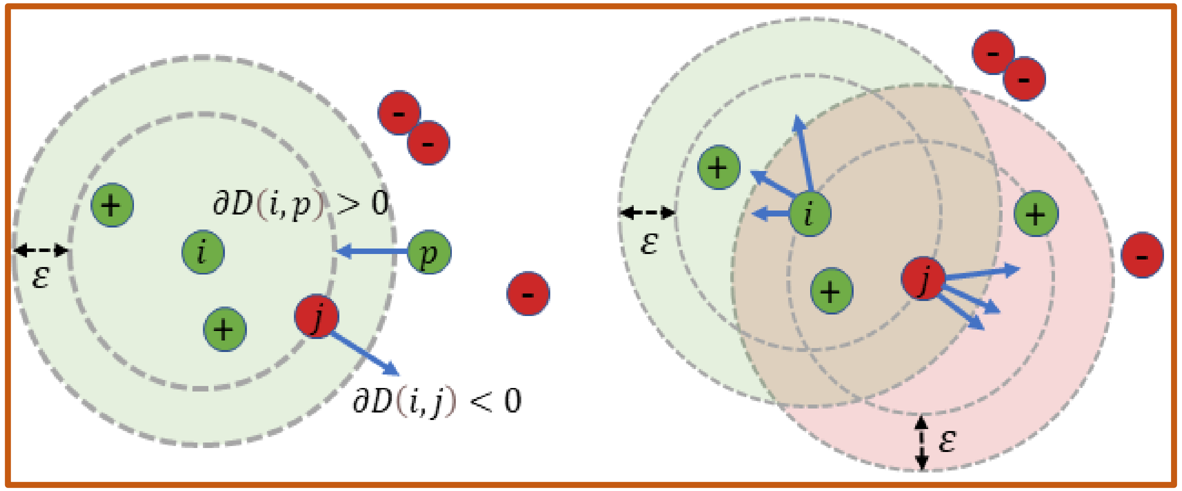
Sum of three losses

- ***Part 1:*** *Contextual similarity optimization*
  - Pull apart contexts of dissimilar samples and push together contexts of similar samples
  - Context means the set of closest neighbors to a sample
- ***Part 2:*** *Similarity regularization*
  - Minimize difference between average similarity of all pairs and a fixed value
- ***Part 3:*** *Standard fixed margin contrastive loss*

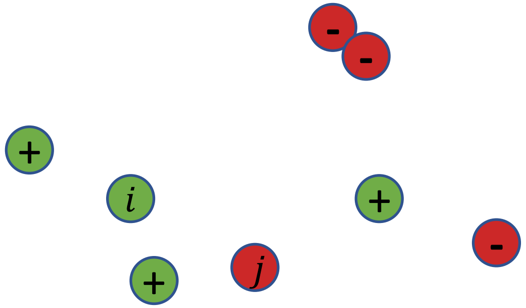
$$\mathcal{L}_{\text{ours}} = \boxed{\lambda\mathcal{L}_{\text{context}}} + (1-\lambda)\mathcal{L}_{\text{contrast}} + \gamma\mathcal{L}_{\text{reg}}$$

(1) Optimize ranking

(2) Optimize intersection of neighborhood sets



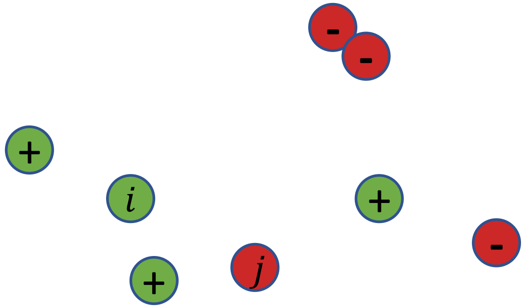
How to Calculate the contextual loss



$$\mathcal{L}_{\text{context}} = \frac{1}{n^2} \sum_{i,j|i \neq j} (y_{ij} - w_{ij})^2$$

Truth: 1 if same label, 0 otherwise.

How to Calculate the contextual loss



Contextual similarity: calculated based on cosine similarity matrix.

$$\mathcal{L}_{\text{context}} = \frac{1}{n^2} \sum_{i,j|i \neq j} (y_{ij} - w_{ij})^2$$

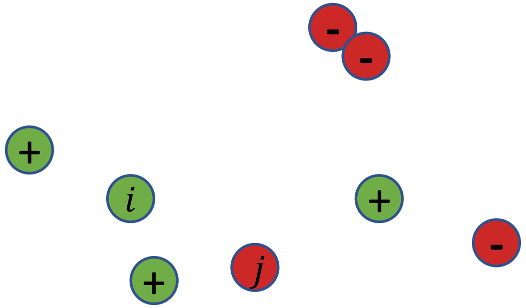
Truth: 1 if same label, 0 otherwise.

How to Calculate the contextual loss

Contextual similarity: calculated based on cosine similarity matrix.

$$\mathcal{L}_{\text{context}} = \frac{1}{n^2} \sum_{i,j|i\neq j} (y_{ij} - w_{ij})^2$$

Truth: 1 if same label, 0 otherwise.



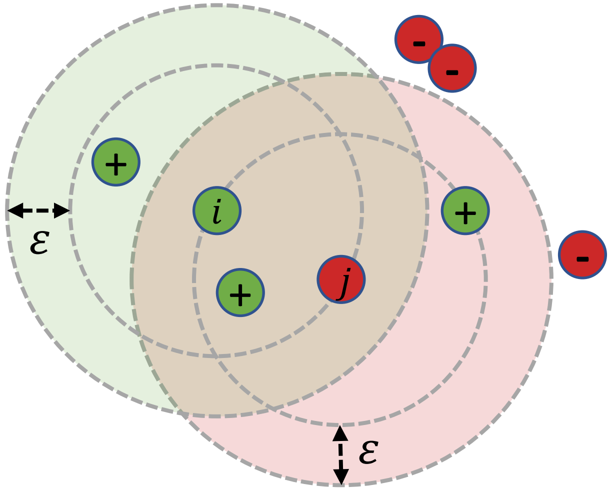
- The contextual similarity captures neighborhood relations.
- The contextual similarity matrix is a function of the cosine similarity matrix.
- The contextual similarity is a number between 0 and 1 predicting the true similarity.
- Example (left): Calculate contextual similarity between *i* and *j*. Two classes per batch, 4 samples per class.

How to Calculate the contextual loss

Contextual similarity: calculated based on cosine similarity matrix.

$$\mathcal{L}_{\text{context}} = \frac{1}{n^2} \sum_{i,j|i \neq j} (y_{ij} - w_{ij})^2$$

Truth: 1 if same label, 0 otherwise.

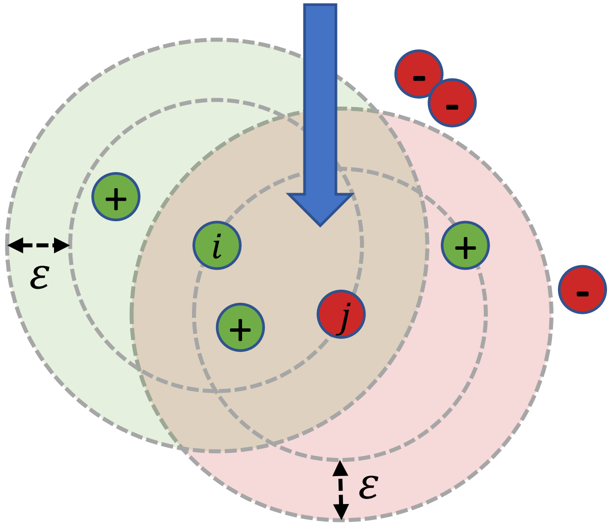


Shading represents neighborhood set.

- The contextual similarity captures neighborhood relations.
- The contextual similarity matrix is a function of the cosine similarity matrix.
- The contextual similarity is a number between 0 and 1 predicting the true similarity.
- Example (left): Calculate contextual similarity between *i* and *j*. Two classes per batch, 4 samples per class.

How to Calculate the contextual loss

i and j have 3 out of 4
neighbors in common.
So contextual similarity = 3/4



Shading represents neighborhood set.

Contextual similarity: calculated based on cosine similarity matrix.

$$\mathcal{L}_{\text{context}} = \frac{1}{n^2} \sum_{i,j|i \neq j} (y_{ij} - w_{ij})^2$$

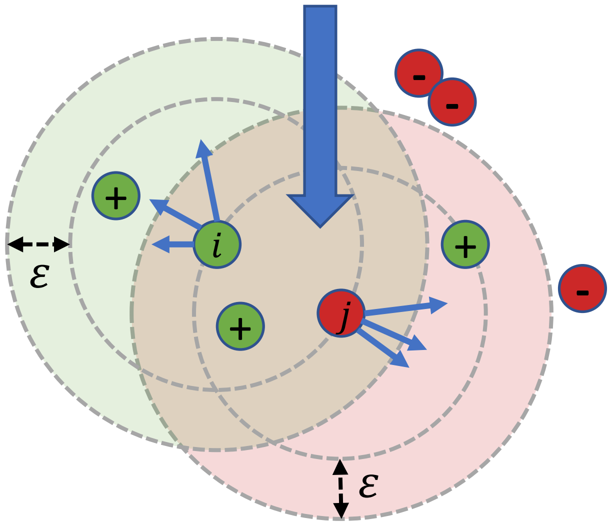
Truth: 1 if same label, 0 otherwise.

- The contextual similarity captures neighborhood relations.
- The contextual similarity matrix is a function of the cosine similarity matrix.
- The contextual similarity is a number between 0 and 1 predicting the true similarity.
- Example (left): Calculate contextual similarity between *i* and *j*. Two classes per batch, 4 samples per class.

How to Calculate the contextual loss

Contextual similarity: calculated based on cosine similarity matrix.

i and j have 3 out of 4 neighbors in common.
So contextual similarity = 3/4



Shading represents neighborhood set.

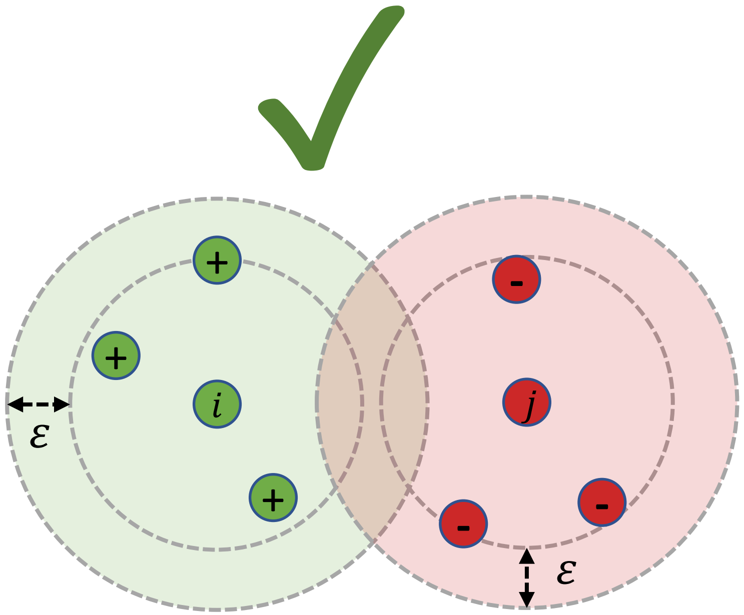
$$\mathcal{L}_{\text{context}} = \frac{1}{n^2} \sum_{i,j|i \neq j} (y_{ij} - w_{ij})^2$$

Truth: 1 if same label, 0 otherwise.

- The contextual similarity captures neighborhood relations.
- The contextual similarity matrix is a function of the cosine similarity matrix.
- The contextual similarity is a number between 0 and 1 predicting the true similarity.
- Example (left): Calculate contextual similarity between *i* and *j*. Two classes per batch, 4 samples per class.

How to Calculate the contextual loss

Contextual similarity: calculated based on cosine similarity matrix.



$$\mathcal{L}_{\text{context}} = \frac{1}{n^2} \sum_{i,j|i \neq j} (y_{ij} - w_{ij})^2$$

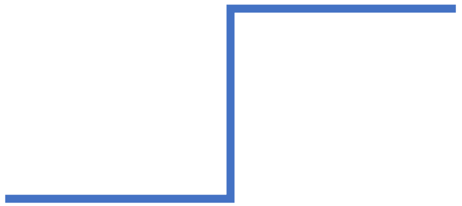
Truth: 1 if same label, 0 otherwise.

- The contextual similarity captures neighborhood relations.
- The contextual similarity matrix is a function of the cosine similarity matrix.
- The contextual similarity is a number between 0 and 1 predicting the true similarity.
- Example (left): Calculate contextual similarity between *i* and *j*. Two classes per batch, 4 samples per class.

Implementation of contextual loss in PyTorch

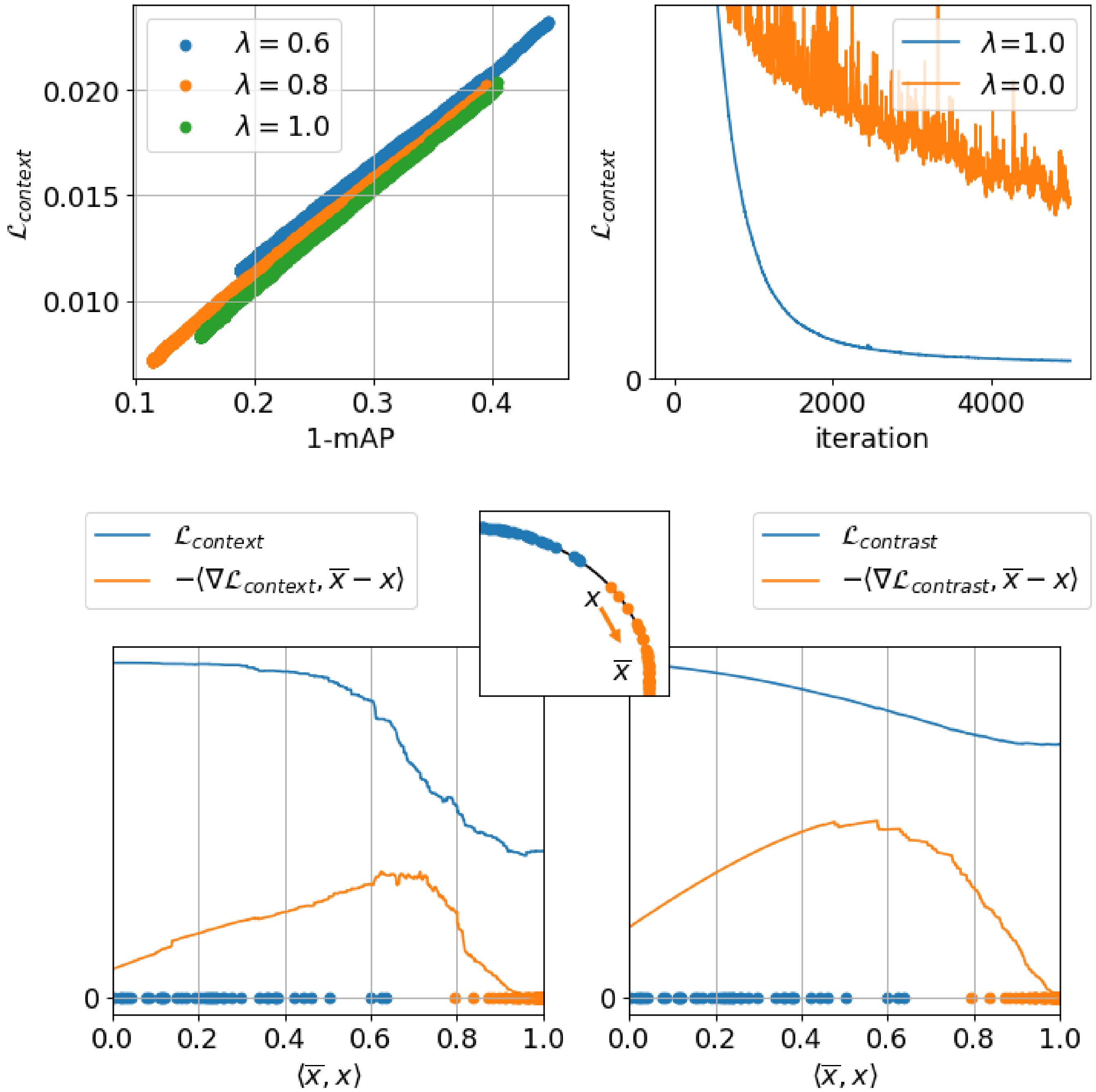
```
class GreaterThan(autograd.Function):
    # Implements theta with artifact gradient
    def forward(x, y):
        return (x >= y).float()
    def backward(g): # Returns gradient w.r.t (x, y)
        return g * alpha, - g * alpha

def get_contextual_similarity(s, k, eps):
    D = 2 - 2 * s                          # Squared Euclidean distance
    Dk = -(-D).topk(k).values[:,-1:] # Distance to k-th neighbor

    Nk_mask = GreaterThan(-D + eps, -Dk.detach())
    M_plus = (Nk_mask @ Nk_mask.T) / Nk_mask.sum(dim=1).detach()
    Nk_mask_not = 1 - Nk_mask
    M_minus = (Nk_mask_not @ Nk_mask_not.T)
                                    / Nk_mask_not.sum(dim=1).detach()
    W_1 = 0.5 * (M_plus + M_minus) * Nk_mask

    # Distance to k/2-th neighbor
    Dk_over_2 = -(-D).topk(k//2).values[:,-1:]
    Nk_over_2_mask = GreaterThan(-D + eps, -Dk_over_2.detach())
    Rk_over_2_mask = Nk_over_2_mask * Nk_over_2_mask.T
    W_2 = (Rk_over_2_mask @ W_1) / Rk_over_2_mask.sum(dim=1)

    return 0.5 * (W_2 + W_2.T)
```

Workaround for undifferentiable heaviside



Sanity Check

- Contextual loss value correlates with 1-mAP (a ranking metric)
- Contextual loss converges
- Gradients “look nice”.



Contextual loss is robust and generalizable

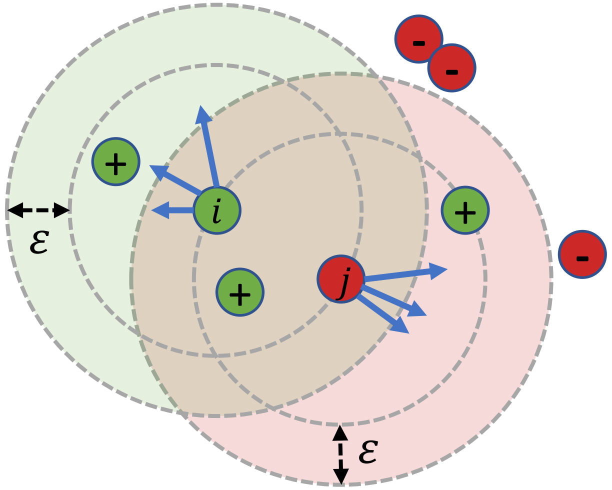


Table 1. Comparison of train and test accuracy on CUB between $\mathcal{L}_{\text{context}}$ and $\mathcal{L}_{\text{context}}$ with gradient corrected according to Eq. 10.

CUB	$\mathcal{L}_{\text{context}}$	with gradient correction
Train R@1	87.0	92.9
Test R@1	71.4	65.4

Contextual loss is robust and generalizable

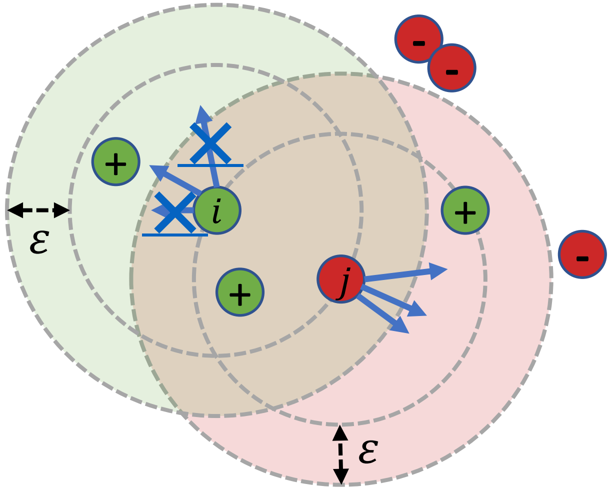
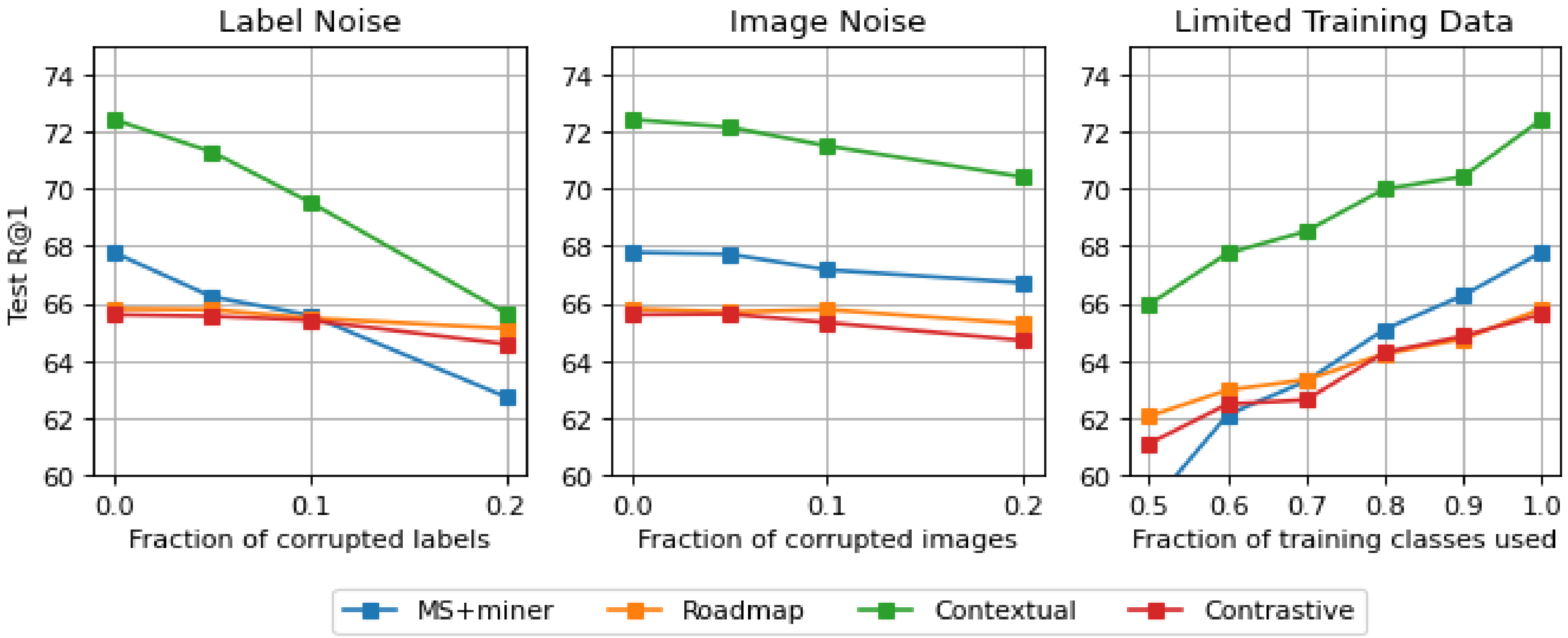


Table 1. Comparison of train and test accuracy on CUB between $\mathcal{L}_{\text{context}}$ and $\mathcal{L}_{\text{context}}$ with gradient corrected according to Eq. 10.

CUB	$\mathcal{L}_{\text{context}}$	with gradient correction
Train R@1	87.0	92.9 ↑
Test R@1	71.4	65.4 ↓

Contextual loss is robust and generalizable (2)



Results

- We achieve strong Recall @ 1 results on four diverse benchmarks
- R @ 1 is the percentage of samples in the test set where the closest test sample has the same label
- R @ k is the percentage of samples in the test set where at least one of the k closest samples has the same label

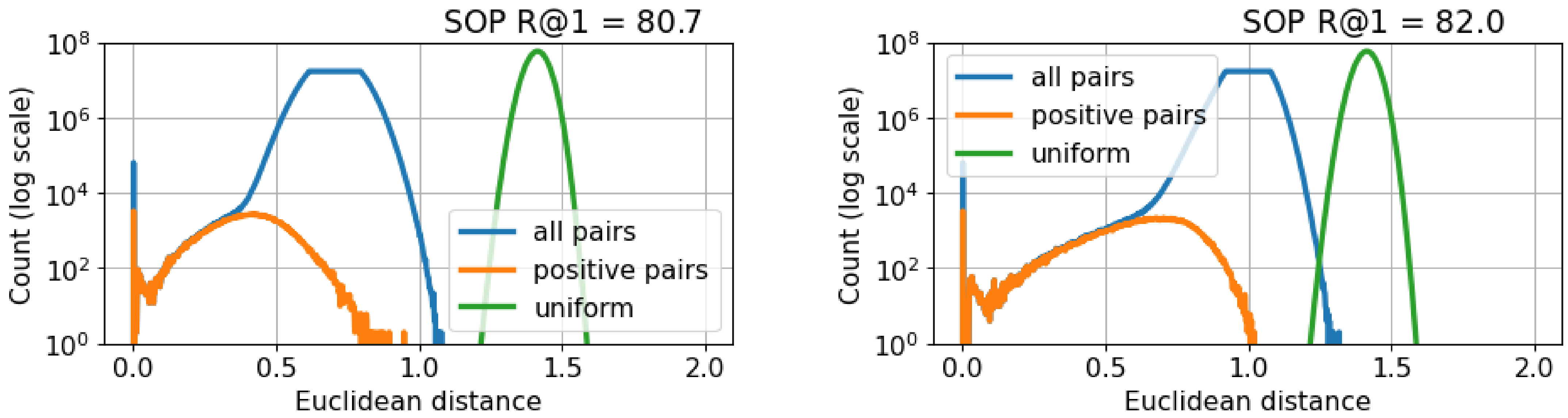
	CUB				Cars			
Method	R@1	R@2	R@4	R@8	R@1	R@2	R@4	R@8
Contrastive	68.5 ± 0.3	78.3 ± 0.1	86.0 ± 0.2	91.3 ± 0.1	85.4 ± 0.2	91.1 ± 0.3	94.6 ± 0.3	96.8 ± 0.1
Triplet	67.3 ± 0.2	77.9 ± 0.1	85.6 ± 0.2	91.2 ± 0.1	77.6 ± 1.3	85.4 ± 0.8	90.8 ± 0.7	94.1 ± 0.4
NtXent	65.7 ± 0.4	76.3 ± 0.2	84.3 ± 0.4	90.0 ± 0.4	79.0 ± 0.6	86.0 ± 0.3	91.0 ± 0.2	94.4 ± 0.3
MS	68.9 ± 0.5	78.5 ± 0.4	86.0 ± 0.6	91.4 ± 0.5	88.7 ± 0.4	93.0 ± 0.2	95.7 ± 0.1	97.3 ± 0.1
N-Softmax†	61.3	73.9	83.5	90.0	84.2	90.4	94.4	96.9
ProxyNCA++†	69.0 ± 0.8	**79.8** ± 0.7	**87.3** ± 0.7	**92.7** ± 0.4	86.5 ± 0.4	92.5 ± 0.3	95.7 ± 0.2	**97.7** ± 0.1
Fast-AP	63.3 ± 0.1	73.7 ± 0.4	82.2 ± 0.3	88.5 ± 0.2	74.7 ± 0.4	82.5 ± 0.7	88.0 ± 0.6	92.2 ± 0.2
Smooth-AP	66.5 ± 0.9	76.6 ± 0.5	84.8 ± 0.6	90.8 ± 0.4	81.1 ± 0.2	87.8 ± 0.4	92.2 ± 0.3	95.1 ± 0.3
ROADMAP	68.7 ± 0.5	78.3 ± 0.3	86.1 ± 0.3	91.1 ± 0.1	84.5 ± 0.5	90.3 ± 0.0	93.9 ± 0.0	96.2 ± 0.1
Ours	**69.8** ± 0.2	**79.8** ± 0.1	**87.1** ± 0.1	92.3 ± 0.2	**89.3** ± 0.0	**93.7** ± 0.2	**96.3** ± 0.1	**97.8** ± 0.2

	SOP			mini-iNaturalist			
Method	R@1	R@10	R@100	R@1	R@4	R@16	R@32
Contrastive	82.4 ± 0.0	91.9 ± 0.0	96.0 ± 0.0	43.5 ± 0.1	62.7 ± 0.1	77.6 ± 0.1	83.2 ± 0.1
Triplet	82.0 ± 0.0	92.5 ± 0.1	96.7 ± 0.0	35.4 ± 0.1	56.5 ± 0.1	74.7 ± 0.1	81.7 ± 0.1
NtXent	79.7 ± 0.2	90.8 ± 0.0	96.1 ± 0.0	40.8 ± 0.1	61.6 ± 0.1	78.0 ± 0.0	83.9 ± 0.0
MS	81.4 ± 0.0	91.4 ± 0.0	96.1 ± 0.1	44.9 ± 0.1	63.9 ± 0.1	78.4 ± 0.1	83.9 ± 0.1
N-Softmax†	78.2	90.6	96.2	–	–	–	–
ProxyNCA++†	80.7 ± 0.5	92.0 ± 0.3	96.7 ± 0.1	–	–	–	–
Fast-AP	80.3 ± 0.1	91.0 ± 0.1	96.0 ± 0.0	35.6 ± 0.2	55.8 ± 0.1	72.8 ± 0.0	79.3 ± 0.0
Smooth-AP	82.0 ± 0.0	92.6 ± 0.0	**96.9** ± 0.0	42.7 ± 0.0	63.3 ± 0.0	79.0 ± 0.0	84.7 ± 0.0
ROADMAP	83.1 ± 0.1	92.6 ± 0.0	96.6 ± 0.0	45.9 ± 0.1	**65.8** ± 0.0	**80.4** ± 0.1	**85.7** ± 0.0
Ours	**83.3** ± 0.0	**92.9** ± 0.1	96.7 ± 0.0	**46.2** ± 0.0	**65.8** ± 0.1	80.2 ± 0.1	85.4 ± 0.1

Similarity Regularization

- ***Problem***: Some metric learning losses (such as ours) tend to make samples close regardless of true similarity
- Undesirable because only a small portion of embedding space is used

- ***Solution***: Regularize average similarity toward a fixed value
- Note most directions in embedding space are orthogonal



Conclusions

- *Contributions*
  - We derive a highly non-trivial differentiable contextual loss function
  - We propose a simple but novel similarity regularizer
  - Our framework improves the state-of-the-art in supervised metric learning in terms of Recall @ 1 accuracy
- *Future work*
  - Investigate theoretical convergence properties of proposed loss function
  - Investigate why similarity regularizer works
  - Extend to multi-label datasets, where similarity score is not binary